CLAIMS

What is claimed is:

1. A method comprising:

receiving a series of operations, an operand of the series of operations being converted from a first format to a second format prior to performing a first operation of the series of operations, a result of the series of operations being converted from the second format to the first format after performing a second operation of the series of operations;

determining the origin of any operands of the series of operations being converted from the first format to the second format; and

if the origin of any operand being converted from the first format to a second format is a conversion from the second format to the first format, then eliminating the redundant conversions from the second format to the first format and from the first format to the second format.

2. The method of claim 1, wherein the series of operations are produced by expanding an application process into basic functions, the basic functions including conversion of operands between the first and second formats.

3. The method of claim 2, further comprising partially compressing the remaining operations after redundant conversions of operands are eliminated.

4. The method of claim 1, wherein the series of operations includes a conversion of the format of an operand that is redundant in a first case and is not redundant in a second case.

5. The method of claim 4, wherein the conversion of the operand that is redundant in the first case and is not redundant in the second case occurs within a loop, and wherein the redundancy is eliminated at least in part by moving one or more operand conversions from inside the loop to outside the loop.

6. The method of claim 1, wherein the first format is a packed floating point format and the second format is an unpacked floating point format.

7. A method comprising:

receiving an application including one or more floating point calculations;

expanding the application by expanding the floating point calculations into a series of floating point operations on floating point operands, the floating point operands being converted from a packed format to an unpacked format prior to the performance of a floating point operation of the series of floating point operations, the result of a floating point operation of the series of floating point operations being converted from the unpacked format to the packed format;

determining the origin of the floating point operands; and

if the origin of any of the floating point operands is a conversion from the unpacked format to the packed format, then eliminating the redundant conversions of the operand from the unpacked format to the packed format and from the packed format to the unpacked format.

8. The method of claim 7, wherein the series of floating point operations are performed by a device that performs software emulation of floating point operations.

9. The method of claim 7, wherein the packed format is a standard floating point format.

10. The method of claim 9, wherein the standard floating point format is in accordance with Standard 754 of the Institute of Electrical and Electronic Engineers.

11. The method of claim 7, wherein the unpacked format is an operand format that may be used in performing floating point emulation of a floating point operation.

12. The method of claim 11, wherein converting a floating point operand from the packed format to the unpacked format comprises converting the floating point operand to one or more integer values.

13. The method of claim 7, further comprising partially compressing the remaining floating point operations, if any, after redundant conversions of operands are eliminated.

14. The method of claim 7, wherein the series of floating point operations includes a conversion of the format of a floating point operand that is redundant in a first case and is not redundant in a second case.

15. The method of claim 14, wherein the conversion of the floating point operand that is redundant in the first case and is not redundant in the second case occurs within a loop, and wherein the redundancy is eliminated at least in part by moving one or more operand conversions from inside the loop to outside the loop.

16. A machine-readable medium having stored thereon data representing sequences of instructions that, when executed by a processor, cause the processor to perform operations comprising:

receiving a series of operations, an operand of the series of operations being converted from a first format to a second format prior to performing a first operation of the series of operations, a result of the series of operations being converted from the second format to the first format after performing a second operation of the series of operations;

determining the origin of any operands of the series of operations being converted from the first format to the second format; and

if the origin of any operand being converted from the first format to a second format is a conversion from the second format to the first format, then eliminating the redundant conversions from the second format to the first format and from the first format to the second format.

17. The medium of claim 16, wherein the series of operations are produced by expanding an application process into basic functions, the basic functions including conversion of operands between the first and second formats.

18. The medium of claim 17, wherein the sequences of instructions include instructions that, when executed by a processor, cause the processor to perform operations comprising partially compressing the remaining operations after redundant conversions of operands are eliminated.

19. The medium of claim 16, wherein the series of operations includes a conversion of the format of an operand that is redundant in a first case and is not redundant in a second case.

20. The medium of claim 19, wherein the conversion of the operand that is redundant in the first case and is not redundant in the second case occurs within a loop, and wherein the redundancy is eliminated at least in part by moving one or more operand conversions from inside the loop to outside the loop.

21. The medium of claim 16, wherein the first format is a packed floating point format and the second format is an unpacked floating point format.

22. A machine-readable medium having stored thereon data representing sequences of instructions that, when executed by a processor, cause the processor to perform operations comprising:

receiving an application including one or more floating point calculations;

expanding the application by expanding the floating point calculations into a series of floating point operations on floating point operands, the floating point operands being converted from a packed format to an unpacked format prior to the performance of a floating point operation of the series of floating point operations, the result of a floating point operation of the series of floating point operations being converted from the unpacked format to the packed format;

determining the origin of the floating point operands; and

if the origin of any of the floating point operands is a conversion from the unpacked format to the packed format, then eliminating the redundant conversions of the operand from the unpacked format to the packed format and from the packed format to the unpacked format.

23. The medium of claim 22, wherein the series of floating point operations are performed by a device that performs software emulation of floating point operations.

24. The medium of claim 22, wherein the packed format is a standard floating point format.

25. The medium of claim 24, wherein the standard floating point format is in accordance with Standard 754 of the Institute of Electrical and Electronic Engineers.

26. The medium of claim 22, wherein the unpacked format is an operand format that may be used in performing floating point emulation of a floating point operation.

27. The medium of claim 26, wherein converting a floating point operand from the packed format to the unpacked format comprises converting the floating point operand to one or more integer values.

28. The medium of claim 22, wherein the sequences of instructions include instructions that, when executed by a processor, cause the processor to perform operations comprising partially compressing the remaining floating point operations, if any, after redundant conversions of operands are eliminated.

29. The medium of claim 22 wherein the series of floating point operations includes a conversion of the format of a floating point operand that is redundant in a first case and is not redundant in a second case.

30. The medium of claim 29, wherein the conversion of the floating point operand that is redundant in the first case and is not redundant in the second case occurs within a loop, and wherein the redundancy is eliminated at least in part by moving one or more operand conversions from inside the loop to outside the loop.